BITS F464 MACHINE LEARNING

Assignment-1

Group details:

Dhairya Agrawal (2020A7PS0130H)

Pulkit Agrawal (2020A7PS2072H)

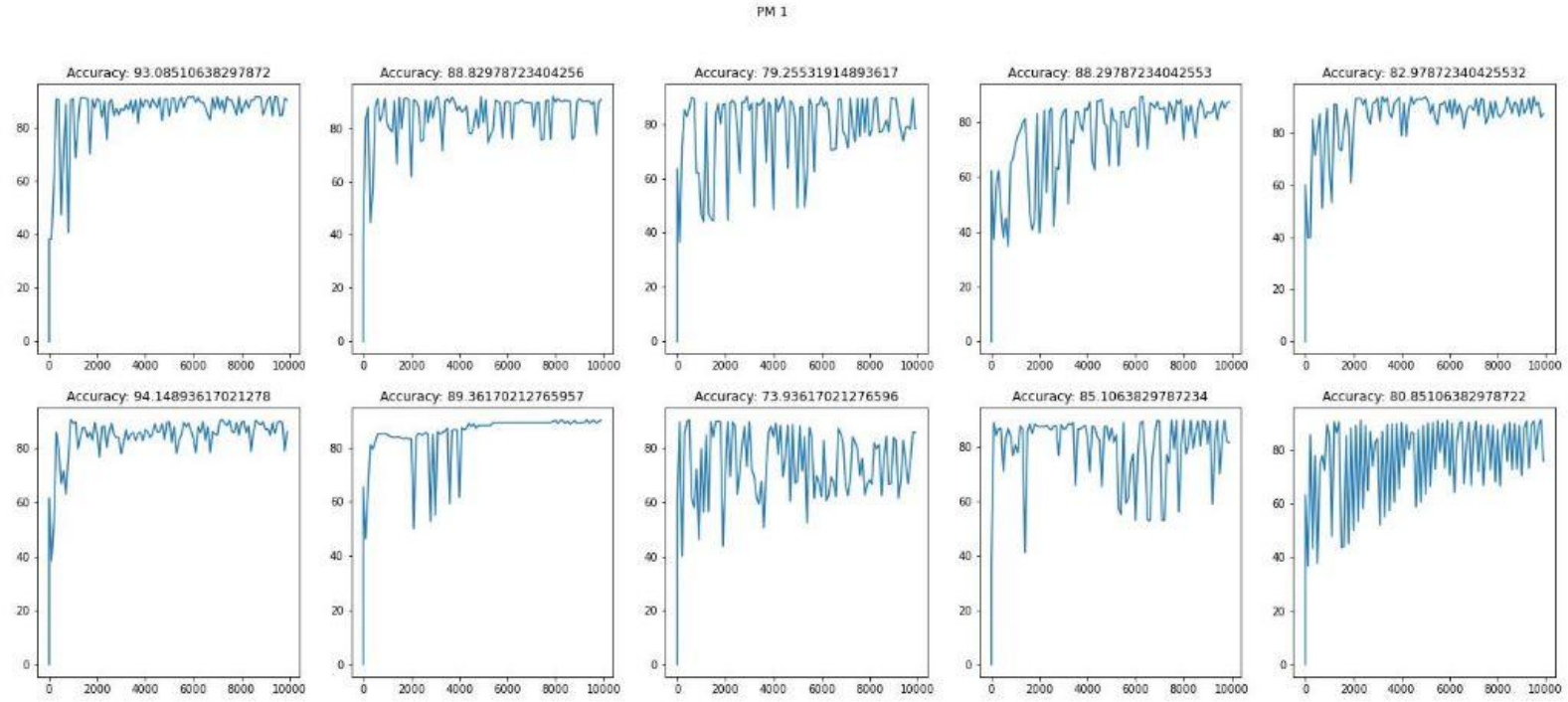
Mohit Agarwal (2020A7PS0189H)

PART A-Perceptron Learning Algorithm

Learning task 1:

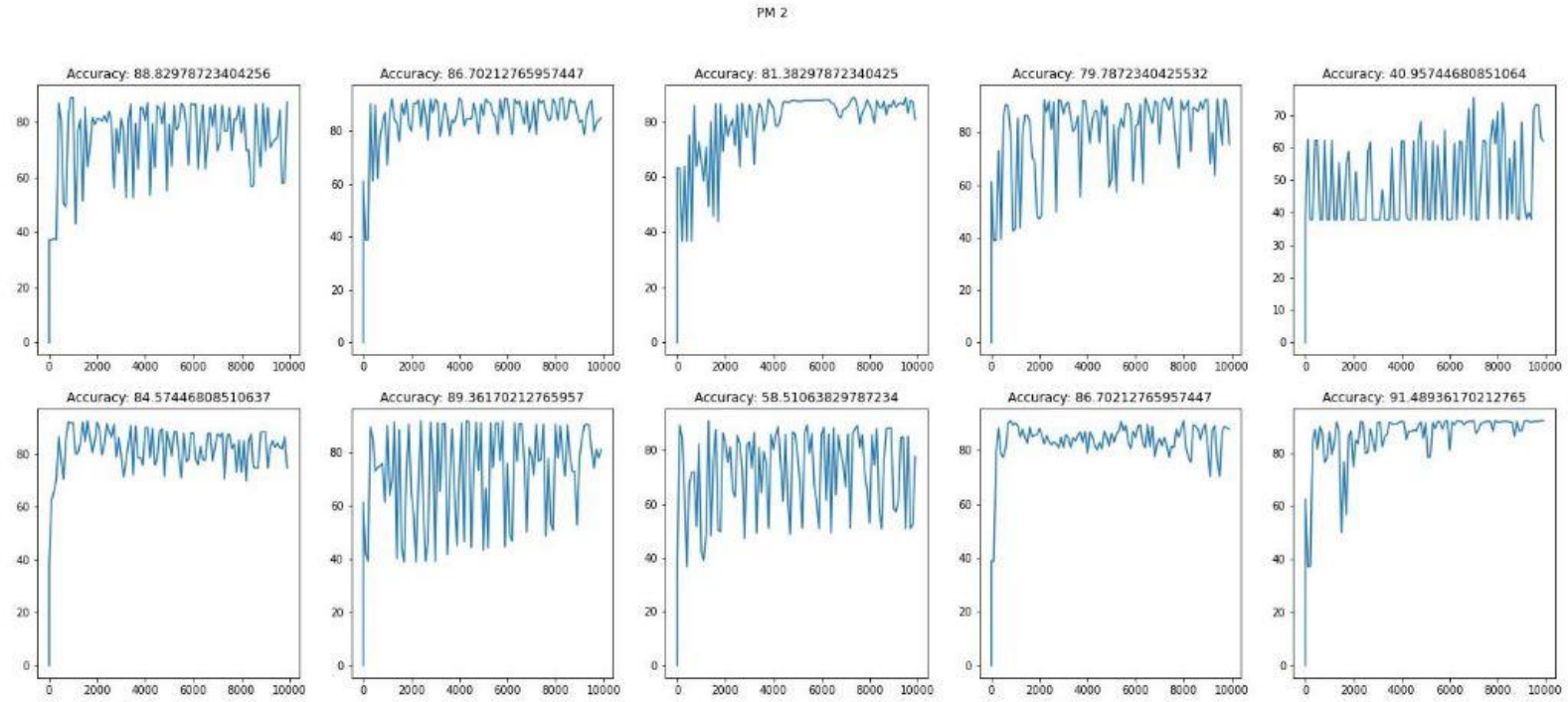
The given task was to build a classifier PM-1 using the perceptron algorithm taught in class and to determine whether the given dataset is linearly separable, then build another classifier PM-2 by changing the order of training examples.

For the PM1, we got the following accuracy:




```
FOR PM1
 Average Accuracy: 85.58510638297874
 Variance in Accuracy: 36.69363965595295
```

For the shuffled data, we created the model PM2 and got the following accuracy:



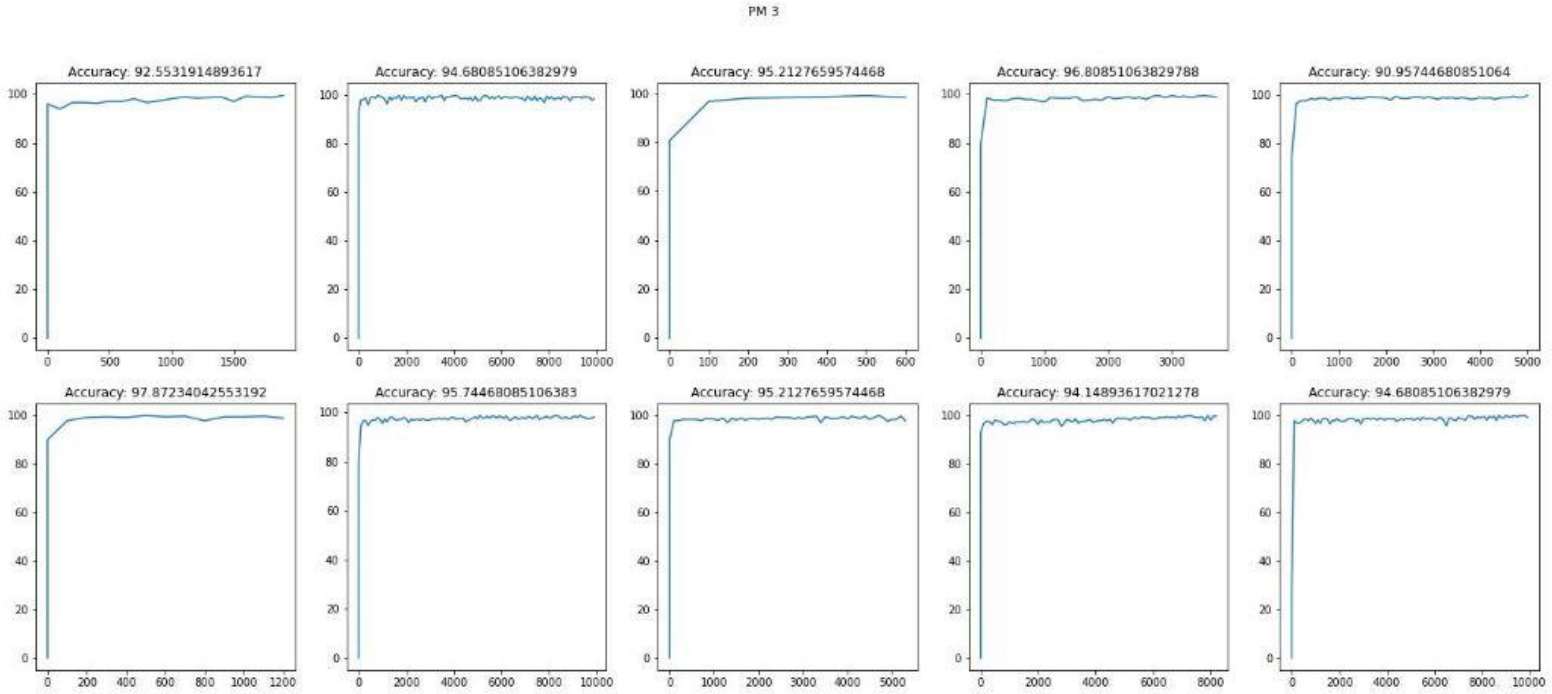

```
FOR PM2
 Average Accuracy: 78.82978723404254
 Variance in Accuracy: 238.2752376641014
```

From the above snippets, the accuracy of the perceptron models changes when we shuffle the data. This happens because the data is not linearly separable, and for a finite number of iterations, we obtain a different decision boundary to separate the points. Hence the accuracies are different.

Learning Task 2:

The given task was to build a classifier PM-3 using the perceptron algorithm on the normalized data and then compare the model PM-1 and PM-3

For the normalized data, we got the following accuracy:




```
FOR PM3
 Average Accuracy: 94.7872340425532
 Variance in Accuracy: 3.4970574920778668
```

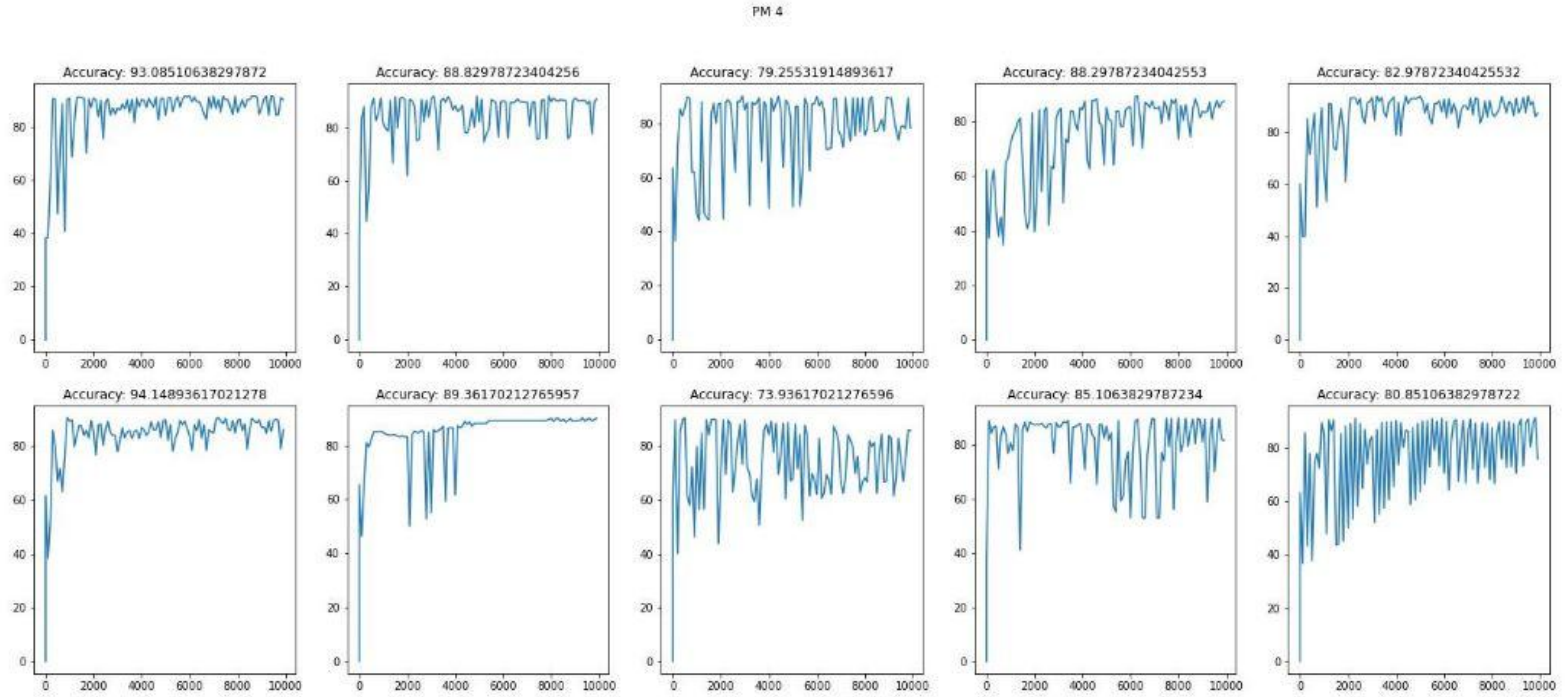
For normalized data, we obtain better results since normalization handles the issues related to scaling and variance in data. We have 30 features in the data with varying scales; hence, normalizing helps us find a better decision boundary than the one we found earlier in PM-1. It also considers the problem where one feature may influence the weights more than others. Overall, it makes the training more stable and prevents the model from being biased toward certain features.

There are no sudden spikes in accuracy since a single feature is not dominating the weights or the decision boundary.

Learning task 3:

The given task required us to change the order of features of the data set randomly and build another model PM-4.

For the shuffled features, we got the following accuracy:




```
FOR PM4
 Average Accuracy: 85.58510638297874
 Variance in Accuracy: 36.69363965595295
```

It can be seen from the above figure and accuracies that the results from PM-1 and PM-4 are identical. This is because changing the order of features does not affect the perceptron model. We can understand this with an example as well. Let us

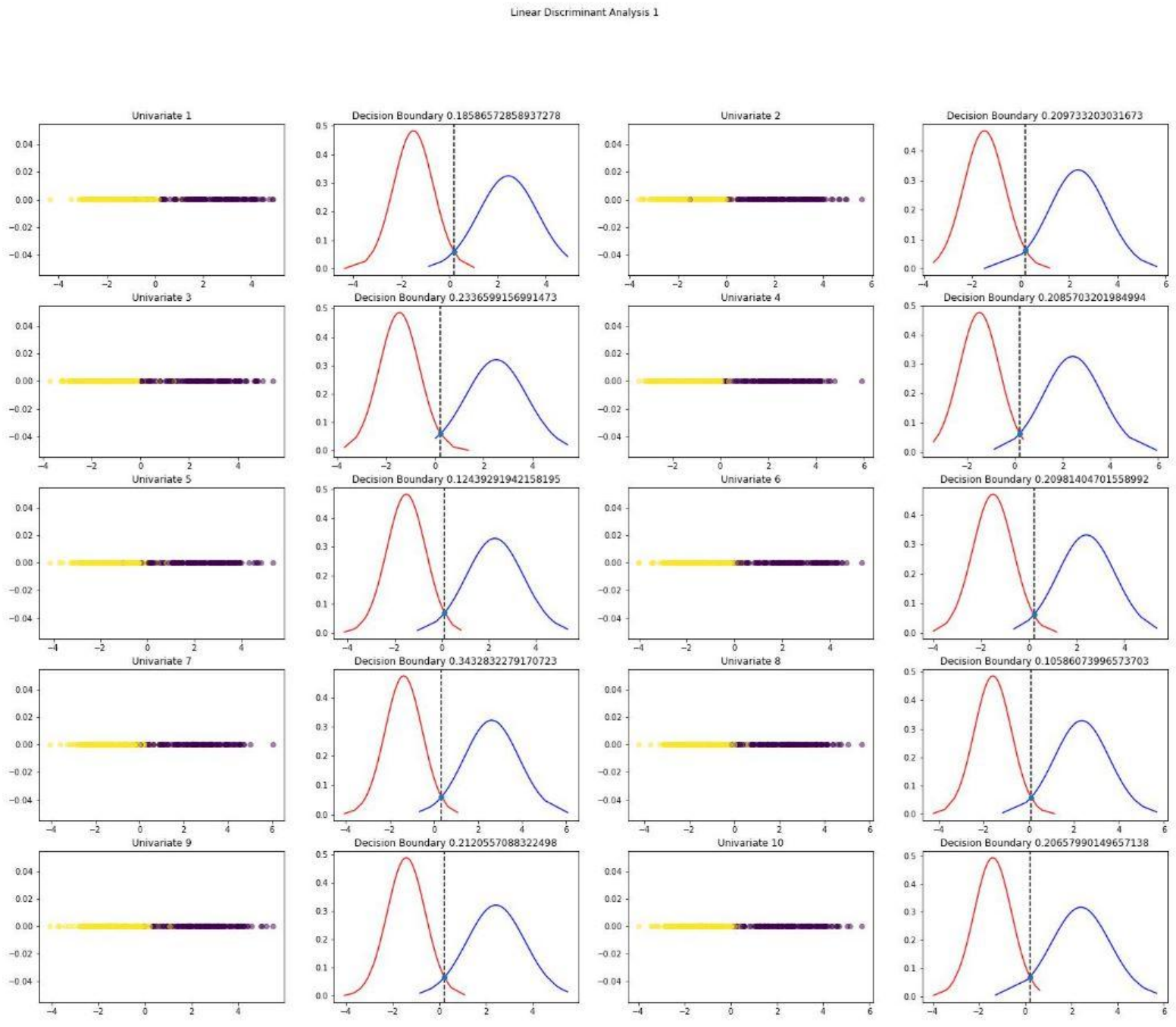
assume that there are only two features, and we can visualize the points on a 2D plane. Now if we change the order and visualize the points as (y, x), we can see that it is only a mirror image of the previous visualization. Therefore even for 30 features, there is no change in the results of PM-1 and PM-4 since even in a 30-dimensional plane, changing the order of planes does not change the performance of the perceptron model. Also, since we change the weights for all features simultaneously, changing the order in which the features are added in a linear summation does not change how weights are updated. Therefore there is no difference between the performance of PM-1 and PM-4.

Part B – Fisher's Linear Discriminant Analysis

Learning Task 1:

The given task required us to build a Fisher's Linear Discriminant Model - FLDM1 on the given training data and reduce the given 32 dimensions to univariate dimension, and find out the decision boundary in the univariate dimension using generative approach.

We got the following univariate plot and decision boundary for the given training data:



After applying the fisher linear discriminant, we converge 32 dimensions into single dimensions, and we get a point alpha(decision boundary) using a generative approach that separates the two classes. From the given dataset after several random training sets, the alphas(decision boundary) are very close to each other with good average accuracy = 96.49%, precision = 96.31%, recall = 98.26%, and F1 score = 97.27 and average variance in accuracy = 0.35, precision = 1.21, recall = 0.68 and F1 score = 0.20.

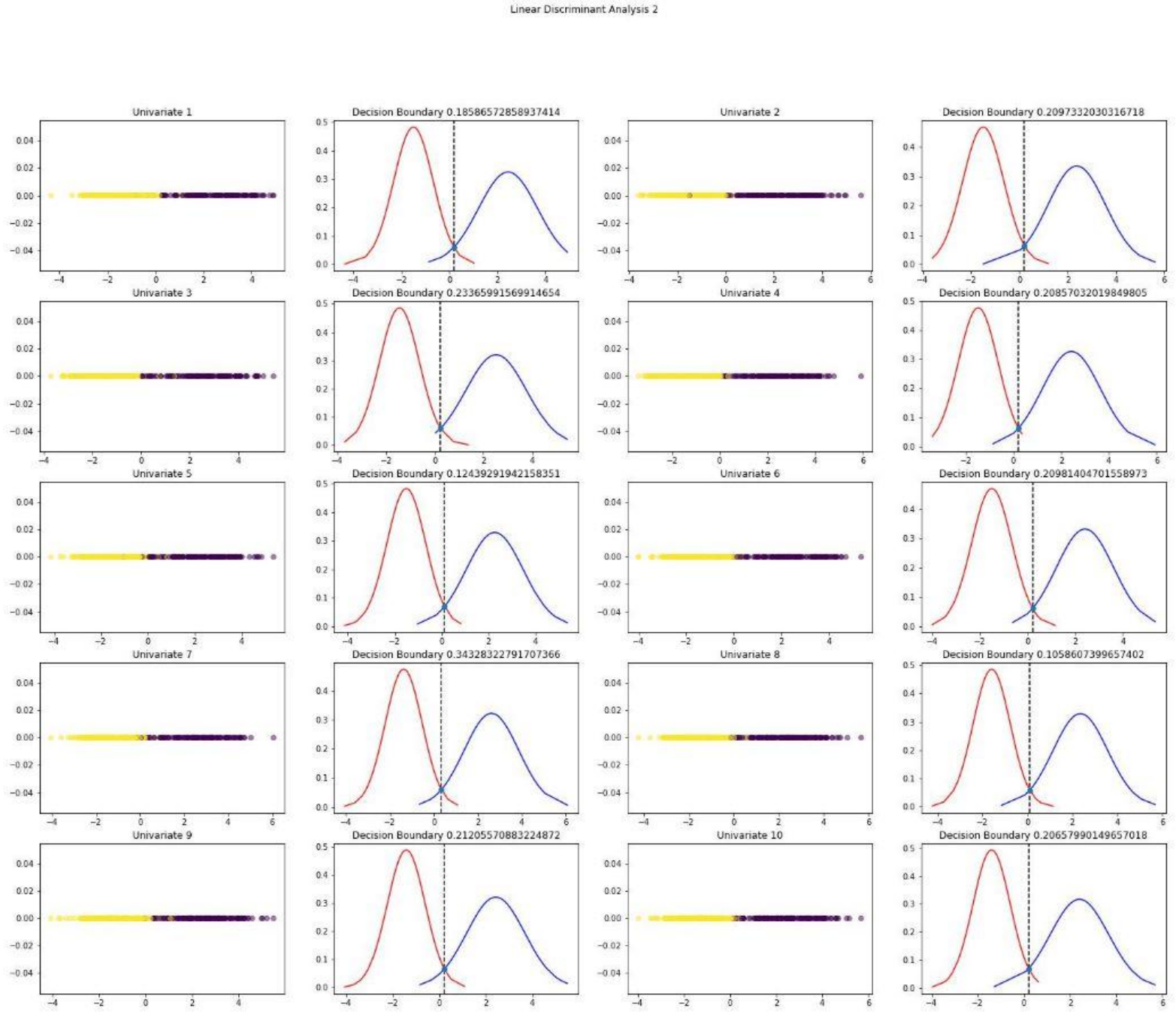
We got the following accuracy on the testing data:

```
For LDA 1 Iteration 1:
Accuracy: 95.74468085106383, Precision: 95.9349593495935, Recall: 97.52066115702479, F1 Score: 96.72131147540983
For LDA 1 Iteration 2:
Accuracy: 95.74468085106383, Precision: 96.0, Recall: 97.5609756097561, F1 Score: 96.7741935483871
For LDA 1 Iteration 3:
Accuracy: 96.80851063829788, Precision: 95.04132231404958, Recall: 100.0, F1 Score: 97.45762711864406
For LDA 1 Iteration 4:
Accuracy: 96.80851063829788, Precision: 96.72131147540983, Recall: 98.33333333333333, F1 Score: 97.5206611570248
For LDA 1 Iteration 5:
Accuracy: 97.34042553191489, Precision: 99.2, Recall: 96.875, F1 Score: 98.02371541501977
For LDA 1 Iteration 6:
Accuracy: 96.27659574468085, Precision: 96.0, Recall: 98.36065573770492, F1 Score: 97.16599190283401
For LDA 1 Iteration 7:
Accuracy: 96.27659574468085, Precision: 95.45454545454545, Recall: 98.13084112149532, F1 Score: 96.77419354838709
For LDA 1 Iteration 8:
Accuracy: 95.74468085106383, Precision: 95.48872180451127, Recall: 98.44961240310077, F1 Score: 96.94656488549617
For LDA 1 Iteration 9:
Accuracy: 96.80851063829788, Precision: 96.63865546218487, Recall: 98.2905982905983, F1 Score: 97.45762711864407
For LDA 1 Iteration 10:
Accuracy: 97.34042553191489, Precision: 96.63865546218487, Recall: 99.13793103448276, F1 Score: 97.87234042553192

Average
 Accuracy: 96.48936170212765, Precision: 96.31181713224792, Recall: 98.26596086874962, F1 Score: 97.27142265953788

Variance in
 Accuracy: 0.35083748302399037, Precision: 1.208597184976558, Recall: 0.6820570094637721, F1 Score: 0.19722546919182246
```

Learning Task 2:

The given task required us to perform random changes in the order of features in the given dataset and build another Fisher's model FLDM2 on the same data and find out the decision boundary in the univariate dimension using a generative approach.

We got the following univariate plot for the given training data after random shuffling of features:



From the above plots we can see that the results are quite similar to learning task 1, and no effect can be seen from shuffling the columns of the training data sets. It also shows that after shuffling the training data, projections of 32 features will map to the same dimension line as that before shuffling. This shows that there is no effect of shuffling the features. The following show the average accuracy = 96.49%, precision = 96.31%, recall = 98.26%, and F1 Score = 97.27%, whereas the variance in accuracy = 0.35, precision = 1.20, recall = 0.68 and F1 score = 0.20

We got the following accuracy on the testing data after random shuffling of features:

```
For LDA 2 Iteration 1:
Accuracy: 95.74468085106383, Precision: 95.9349593495935, Recall: 97.52066115702479, F1 Score: 96.72131147540983
For LDA 2 Iteration 2:
Accuracy: 95.74468085106383, Precision: 96.0, Recall: 97.5609756097561, F1 Score: 96.7741935483871
For LDA 2 Iteration 3:
Accuracy: 96.80851063829788, Precision: 95.04132231404958, Recall: 100.0, F1 Score: 97.45762711864406
For LDA 2 Iteration 4:
Accuracy: 96.80851063829788, Precision: 96.72131147540983, Recall: 98.33333333333333, F1 Score: 97.5206611570248
For LDA 2 Iteration 5:
Accuracy: 97.34042553191489, Precision: 99.2, Recall: 96.875, F1 Score: 98.02371541501977
For LDA 2 Iteration 6:
Accuracy: 96.27659574468085, Precision: 96.0, Recall: 98.36065573770492, F1 Score: 97.16599190283401
For LDA 2 Iteration 7:
Accuracy: 96.27659574468085, Precision: 95.45454545454545, Recall: 98.13084112149532, F1 Score: 96.77419354838709
For LDA 2 Iteration 8:
Accuracy: 95.74468085106383, Precision: 95.48872180451127, Recall: 98.44961240310077, F1 Score: 96.94656488549617
For LDA 2 Iteration 9:
Accuracy: 96.80851063829788, Precision: 96.63865546218487, Recall: 98.2905982905983, F1 Score: 97.45762711864407
For LDA 2 Iteration 10:
Accuracy: 97.34042553191489, Precision: 96.63865546218487, Recall: 99.13793103448276, F1 Score: 97.87234042553192

Average
 Accuracy: 96.48936170212765, Precision: 96.31181713224792, Recall: 98.26596086874962, F1 Score: 97.27142265953788

Variance in
 Accuracy: 0.35083748302399037, Precision: 1.208597184976558, Recall: 0.6820570094637721, F1 Score: 0.19722546919182246
```

As also discussed in the PM-3 model, there is no effect of changing the columns since in the end, we project them onto the plane which provides us with a minimum difference of means and maximum separation of variance.

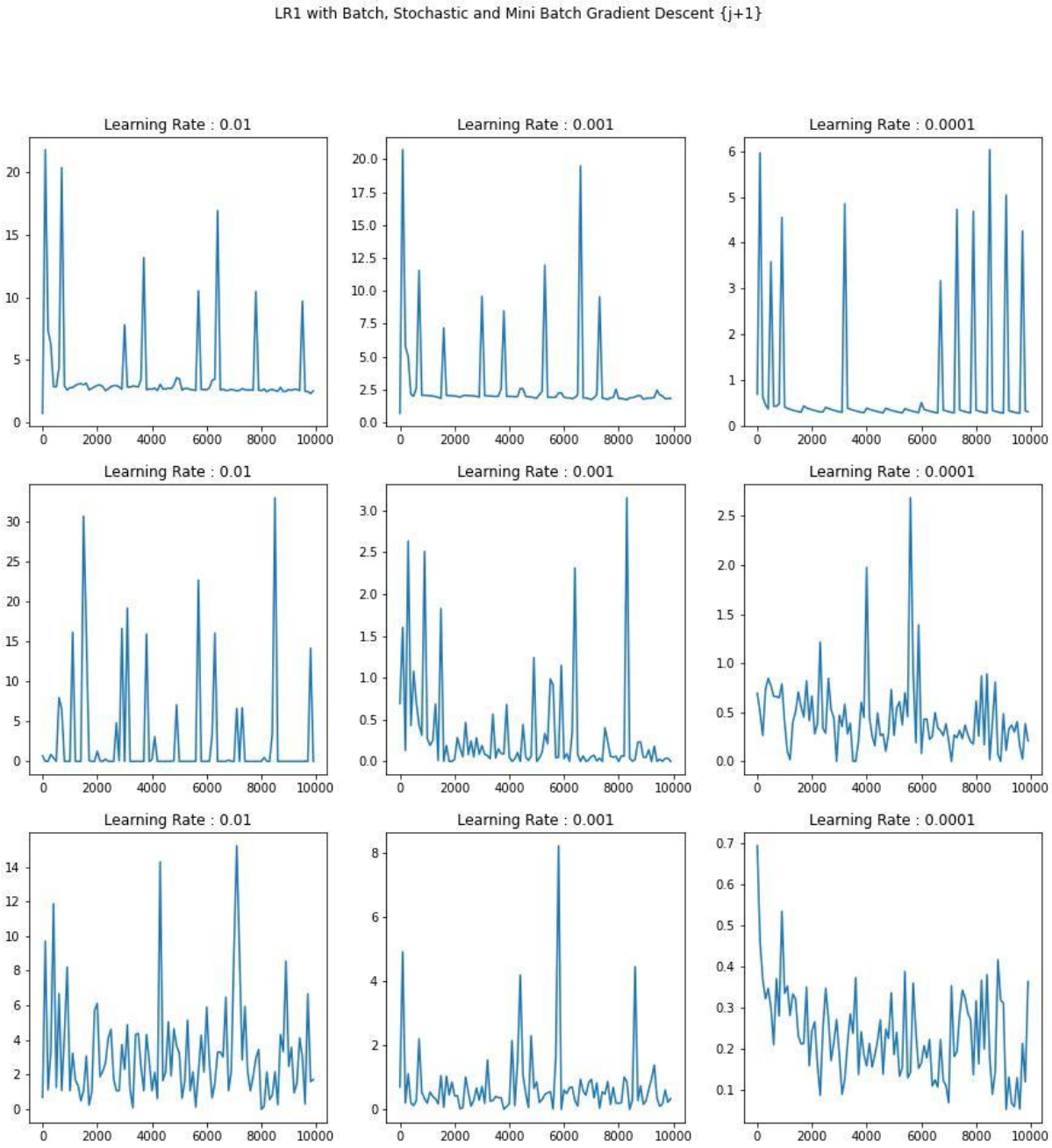
From the above results, we can conclude that changing the order of features does not affect the accuracy of Fisher's Linear Discriminant model .

Part C- Logistic Regression

Learning Task 1:

This task required us to build a Logistic Regression Model LR1 and find the accuracy of testing data with various probability thresholds

Following are the results for Batch, Stochastic, and Mini Batch gradient descents, respectively:



Batch gradient descent: Because we are averaging over all of the training data's gradients for a single step, the cost vs. epochs graph is relatively smooth. As the epochs progress, the cost function keeps decreasing.

Stochastic analysis: Because we only consider one example at a time, the cost will vary over the training examples and may not always decrease. Yet, with time, we notice that the cost function fluctuates less. Also, because the cost fluctuates, it will never reach the minimum, but it will keep revolving around it. Stochastic Gradient Descent applies to larger datasets. Large datasets result in more frequent parameter adjustments, which speeds up convergence.

Mini Batch Gradient: Because we only average a few samples at a time, the mini-batch gradient descent's average cost throughout the epochs is similar to the Stochastic Gradient Descent.

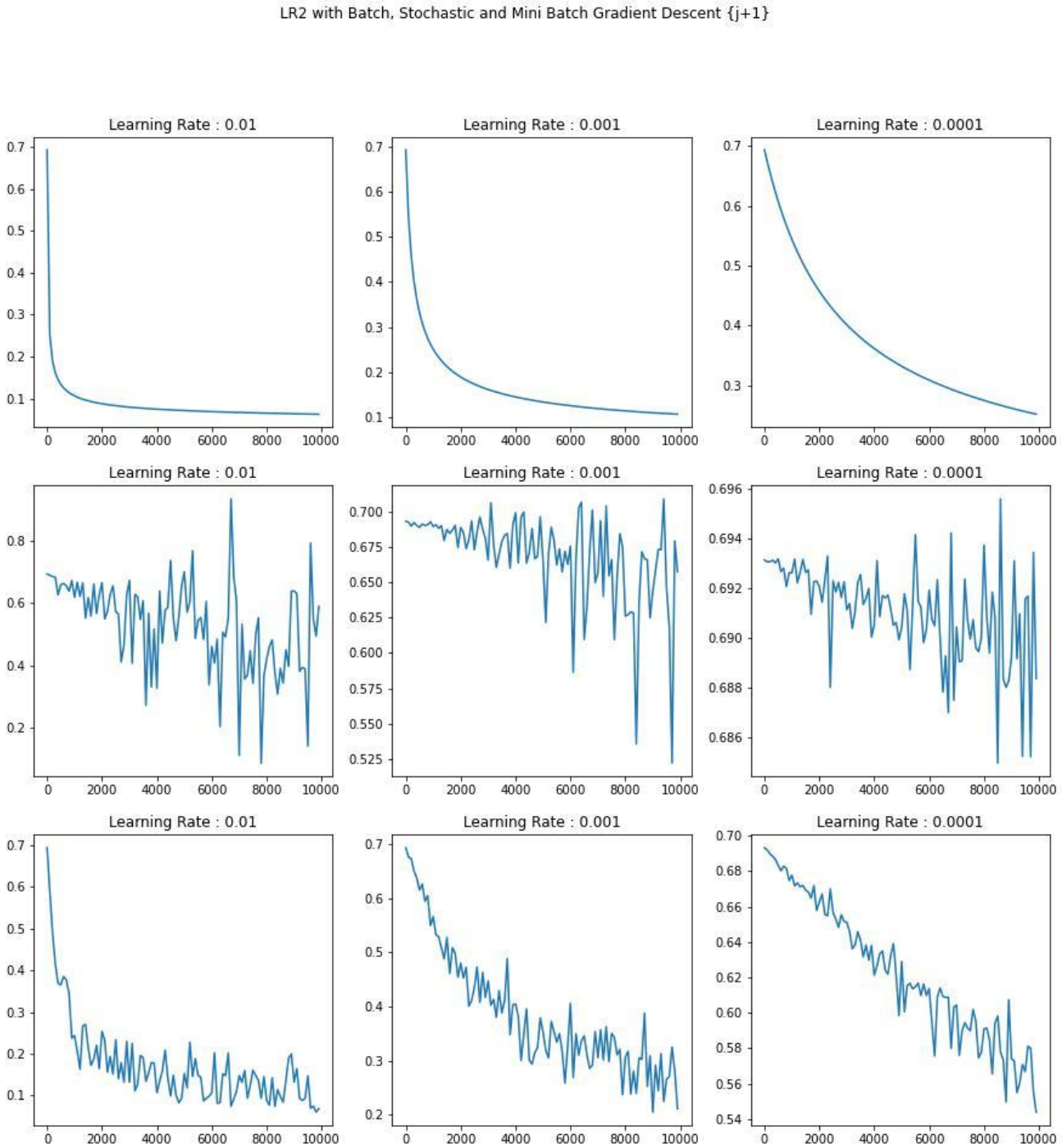
The overall cost vs. iterations vary wildly and may look random, an effect of unnormalized data. Since unnormalized data may contain some features which affect the weights more than others, this results in a bias of weights towards a particular feature which may not be true for the dataset. This is the reason we get huge spikes in cost.

Also, since some features may have high variance, without normalizing the data, those features may tend to dominate the training process and overfit the dataset.

Learning task 2:

The given task required us to build another Logistic Regression model LR2 after applying Feature Engineering Task 1 and Feature Engineering Task 2 with various threshold values and checking the accuracy with testing data

Following are the results for Batch, Stochastic and Mini Batch gradient descents:



Following are the results for Batch, Stochastic, Mini Batch gradient descendants for learning rate 0.01,0.001,0.0001

```
Logistic Regression 2


Batch :
For Learning Rate 0.01
 Average
 Accuracy : 97.9787234042553, Precision : 98.63891535021567, Recall : 98.11965811965811, F1 Score : 98.36214960198325
 Variance in
 Accuracy : 1.0638297872340463, Precision : 0.42181542262595856, Recall : 4.207758053911897, F1 Score : 0.7334278630092262
For Learning Rate 0.001
 Average
 Accuracy : 97.02127659574468, Precision : 97.67994571117859, Recall : 97.60683760683762, F1 Score : 97.59415632709496
 Variance in
 Accuracy : 2.1050248981439545, Precision : 3.6448925673935384, Recall : 8.590839360070134, F1 Score : 1.4629969001429686
For Learning Rate 0.0001
 Average
 Accuracy : 91.38297872340426, Precision : 95.56353402446199, Recall : 90.76923076923077, F1 Score : 92.65292677827797
 Variance in
 Accuracy : 20.30330466274332, Precision : 11.26069216670551, Recall : 95.66805464241367, F1 Score : 19.530979756630064

Stochastic :
For Learning Rate 0.01
 Average
 Accuracy : 80.95744680851064, Precision : 92.16815038591328, Recall : 78.63247863247864, F1 Score : 80.67165326825561
 Variance in
 Accuracy : 186.89452240832952, Precision : 67.37927052811747, Recall : 729.9291401855505, F1 Score : 353.0748559433182
For Learning Rate 0.001
 Average
 Accuracy : 59.042553191489354, Precision : nan, Recall : 58.97435897435897, F1 Score : nan
 Variance in
 Accuracy : 433.2277048438206, Precision : nan, Recall : 2322.15647600263, F1 Score : nan
For Learning Rate 0.0001
 Average
 Accuracy : 58.82978723404254, Precision : nan, Recall : 58.97435897435897, F1 Score : nan
 Variance in
 Accuracy : 431.5980081484834, Precision : nan, Recall : 2322.15647600263, F1 Score : nan

Mini Batch :
For Learning Rate 0.01
 Average
 Accuracy : 96.70212765957447, Precision : 97.52580540042676, Recall : 97.26495726495726, F1 Score : 97.33451958113316
 Variance in
 Accuracy : 2.421910366681753, Precision : 4.729997990626815, Recall : 10.05186646212287, F1 Score : 1.6946087658387718
For Learning Rate 0.001
 Average
 Accuracy : 90.63829787234042, Precision : 95.24138536197526, Recall : 89.91452991452991, F1 Score : 92.00876920300941
 Variance in
 Accuracy : 20.21276595744684, Precision : 13.495419999896114, Recall : 100.05113594857191, F1 Score : 19.910792877714254
For Learning Rate 0.0001
 Average
 Accuracy : 64.46808510638299, Precision : nan, Recall : 62.051282051282044, F1 Score : nan
 Variance in
 Accuracy : 408.71435038478944, Precision : nan, Recall : 1968.471035137702, F1 Score : nan
```

We can see from the above results that the graphs of cost vs. iterations obtained are much cleaner than those obtained from LR-1. This is since now a single feature

does not affect the weights very much but does so only proportionately. It also reduces the scale of the features so that they are comparatively on a similar feature scale.

The benefit of this normalization method comes from the fact that all data associations are retained precisely, and it decreases the impact of outliers on the data.

Normalizing the data can increase the stability of the training process and avoid the model being biased towards particular attributes. This may result in a more effective model when generalized to new data.

The graph trends indicate that the cost decreases gradually with increasing iterations. It is more smooth in the case of batch gradient descent since, for each epoch, we train the weights on the complete data rather than a single example or batch. It is a bit noisy for stochastic models since we only train the weights after seeing a random training example and training on that. Still, it is evident that with increasing iterations, the loss converges.

The data converges faster for large learning rate values, and we find the minimum cost faster but may also overestimate and miss the global minima. This is true for all the gradient descent techniques, batch, stochastic, and mini-batch.

Part D – Comparative Study

Learning task 1:

The task required us to perform a comparative study of PM1, PM3, PM4, FLDM1, FLMD2, LR1, LR2.

The Perceptron algorithm did not converge even after high iterations and didn't give 100% accuracy hence we can conclude that the given dataset was not linearly separable.

We could see that PM1 and PM4 performed exactly sthe ame since changing the order of features in the dataset would not affect the result of the Perceptron algorithm.

PM3 performed better than PM1, PM2 and PM4 because of using normalized data which improves the accuracy for PM3 we got accuracy of 94.7% as we know that normalising the data decreases the impact of outliers of the data. Normalising the data helped to reduce the variance of the features and made training process more stable.

We also saw that FLDM1 and FLDM2 performed exactly same since changing the order of features would not affect the result of Fisher's Linear Discriminant Algorithm.

FLDM gave an average accuracy of 96.4 %

Within LR1 and LR2 we could see that LR2 performed much better and consistent than LR1 since the normalized data gives smoother dataset and avoids the occurrence and formation of spikes

Within LR2 we saw that Batch gradient descent gave best results with threshold value 0.5 an accuracy of 98.4%

From the results we obtained, we conclude that the best results were given by LR2 using Batch Gradient Descent with learning rate 0.01 and 0.02 for threshold 0.5. For higher threshold values there will be more false negatives since even positive data will not be classified correctly. Similarly, for lower threshold values there are more false positives.

From a healthcare dataset perspective, we require lower threshold values since we can not bear to have false negatives, i.e. Cancer patients labeled as not cancer. Therefore we require higher precision.

In general, we can see that the results are always better for normalized data since a single feature cannot influence the weights in perceptron and logistic regression, as well as the projection plan in linear discriminant analysis. Also, using normalized data, the features are scaled, and the training is stable and gives a good performance.